import random as r

def ABPruning(mx, A, B, li, brn, idx, lvl, dpt):

global pr

if dpt == lvl:

pr += 1

return li[idx]

if mx != False:

val = -float("inf")

for c in range(brn):

val = max(val, ABPruning(False, A, B, li, brn, ((idx \* brn) + c), (lvl + 1), dpt))

A = max(A, val)

if B <= A:

break

return val

else:

val = +float("inf")

for c in range(brn):

val = min(val, ABPruning(True, A, B, li, brn, ((idx \* brn) + c), (lvl + 1), dpt))

B = min(B, val)

if A >= B:

break

return val

pr = 0

id = input("Enter your student id : ")

hp\_ren = input("Minimum and Maximum value for the range of negative HP : ")

newHP = int(id[-2:][::-1])

dpt = int(id[0])\*2

hp\_min = int(hp\_ren.split()[0])

hp\_max = int(hp\_ren.split()[1])

brn = int(id[2])

len\_leaf = brn \*\* dpt

#li = [19,22,9,2,26,16,16,27,16]

#li = [18,13,5,12,10,5,13,7,17,8,6,8,5,11,13,18]

li = [r.randint(hp\_min, hp\_max+1) for i in range(len\_leaf)]

dam = ABPruning(True, -float("inf"), +float("inf"), li, brn, 0, 0, dpt)

print("1. Depth and Branches ratio is {}:{}".format(dpt, brn))

print("2. Terminal States(Leaf Nodes) are {}".format(",".join(str(i) for i in li)))

print("3. Left life(HP) of the defender after mx damage caused by the attacker is {}".format(newHP - dam))

print("4. After Alpha-Beta Pruning Leaf Node Comparisons {}".format(pr))